Arquitectura de Computadores

Santiago Ramirez Arenas
Docente: José Alfredo Jaramillo Villegas
Universidad Tecnológica de Pereira

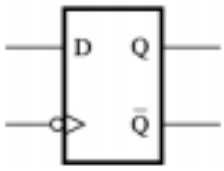
29 de octubre de 2021

Dispositivos de Lógica Secuencial

Objetivo
En este laboratorio se estudiarán los dispositivos de lógica secuencial de escala media de integración (MSI por sus siglas en inglés). Estos dispositivos incluyen flipflops, registros, registros de corrimiento y contadores. Se diseñarán estos dispositivos usando diferentes enfoques haciendo un uso efectivo de las opciones sintácticas que ofrece el lenguaje de descripción de hardware SystemVerilog. Dispositivos de Lógica Secuencial.

En este laboratorio se deberá diseñar y verificar los siguientes dispositivos de lógica secuencial de escala media de integración usando el lenguaje de descripción de hardware SystemVerilog:

Flipflop Tipo D sensible al flanco de bajada de reloj usando una descripción comportamental.



Diseño Flipflop Tipo D sensible al flanco de bajada

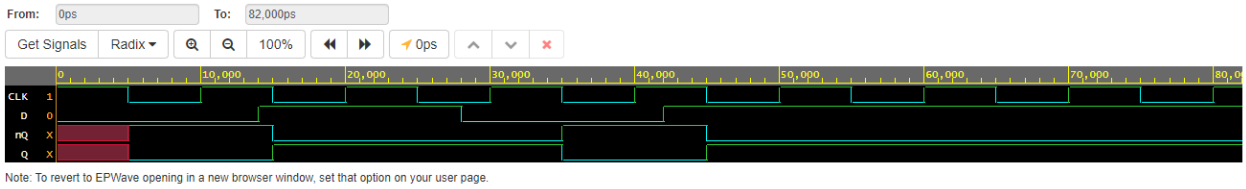
```
1  `timescale  1ns / 10 ps
2  module flipflop_D(
3    input D,
4    output reg Q,
5    output nQ,
6    input CLK);
7
8    assign nQ = ~Q;
9
10   always @(negedge CLK)
```

```
  Q = D ;
endmodule
```

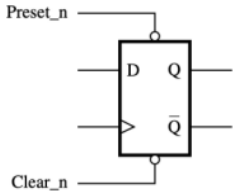
Testbench Flipflop Tipo D sensible al flanco de bajada

```
`timescale  1ns / 10 ps

module test_flipflop_D();
  reg D, CLK;
  wire Q, nQ;

  flipflop_D circuito (D, Q,nQ, CLK);

  initial CLK = 1'b1;
  always #5 CLK = ~CLK;

  initial
    begin
      $dumpfile ("out.vcd");
      $dumpvars (1,test_flipflop_D);

           D = 1'b0;
       #14 D = 1'b1;
       #14 D = 1'b0;
       #14 D = 1'b1;
       #40
      $finish;
    end
endmodule
```

Simulación Flipflop Tipo D sensible al flanco de bajada



Flip-flop Tipo D sensible al flanco de subida de reloj con preset y clear activos en bajo usando una descripción comportamental



Diseño Flip-flop Tipo D sensible al flanco de subida de reloj con preset y clear activos en bajousando una descripción comportamenta

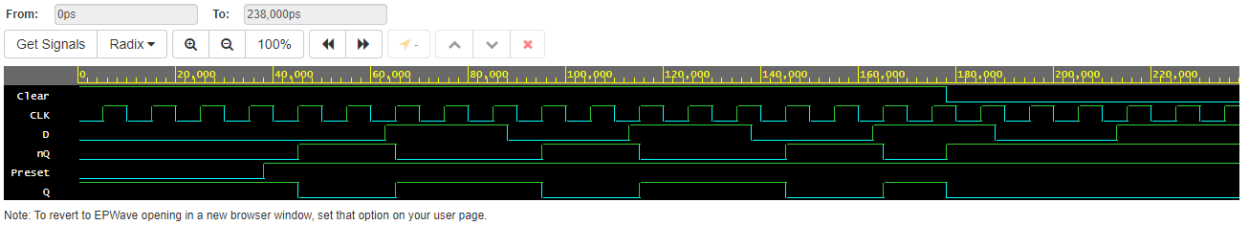
```
‘timescale  1ns / 10 ps

module flipflop_D_Preset_Clear(
  input D,
  output reg Q,
  output nQ,
  input CLK,Preset, Clear);

  assign nQ = ~ Q;

  always @(Preset,Clear)
    if (Preset == 0)
      Q = 1’b1;
  else if(Clear == 0)
    Q = 1’b0;

  always @(posedge CLK)
    if (Preset == 1 && Clear == 1)
      Q = D;
endmodule
```

Testbench Flip-flop Tipo D sensible al flanco de subida de reloj con preset y clear activos en bajousando una descripción comportamental

```
‘timescale  1ns / 10 ps

module test_flipflop_D_Preset_Clear();
  reg D, CLK,Preset,Clear;
  wire Q, nQ;

  flipflop_D_Preset_Clear circuito (D,Q,nQ,CLK,Preset,Clear);

   initial CLK = 1’b0;
   always #5 CLK = ~CLK;

  initial
    begin
      $dumpfile ("out.vcd");
      $dumpvars (1,test_flipflop_D_Preset_Clear);

      D = 1’b0 ; Preset = 1’b0 ; Clear = 1’b1;

      #38 Preset = 1’b1 ; Clear = 1’b1;
      #25 D = 1’b1;
      #25 D = 1’b0;
      #25 D = 1’b1;
      #25 D = 1’b0;
      #25 D = 1’b1;
      #15 Clear = 1’b0;
      #10 D = 1’b0;
      #25 D= 1’b1;
      #25
```

```
    $finish;
     end
endmodule
```

Simulación Flip-flop Tipo D sensible al flanco de subida de reloj con preset y clear activos en bajousando una descripción comportamental



Registro de 4 bits
o Usando una descripción estructural instanciando flip-flops tipo D.
o Usando una descripción comportamental.

Diseño Usando una descripción estructural instanciando flip-flops tipo D

```
  `timescale  1ns / 10 ps

module registro_4bits(
  input CLK,
  output [3:0]Q,
  input In);

  flipflop_D ff0 (In,Q[3],CLK);
  flipflop_D ff1 (Q[3],Q[2],CLK);
  flipflop_D ff2 (Q[2],Q[1],CLK);
  flipflop_D ff3 (Q[1],Q[0],CLK);

endmodule
```

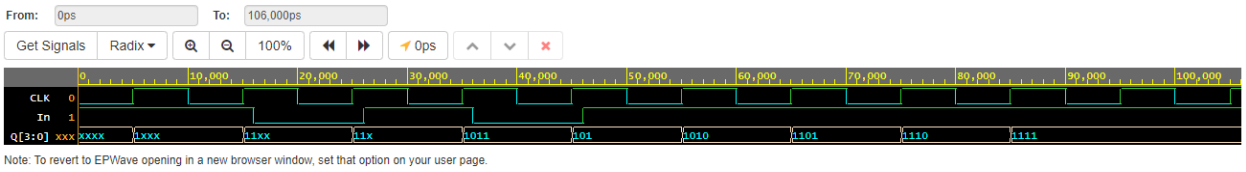
Diseño flip-flop tipo D

```
`timescale  1ns / 10 ps

module flipflop_D(
  input D,
  output reg Q,
  input CLK);

  always @(posedge CLK)
    Q = D;
endmodule
```

Testbench Usando una descripción estructural instanciando flip-flops tipo D

```
‘timescale  1ns / 10 ps

module test_registro_4bits ();
  reg CLK,In;
  wire [3:0] Q;

  registro_4bits circuito (CLK,Q,In);

  initial CLK = 1’b0;
  always #5 CLK = ~CLK;

  initial
    begin
      $dumpfile ("out.vcd");
      $dumpvars (1,test_registro_4bits);

      In =1’b1;
      #16 In= 1’b0;
      #10 In= 1’b1;
      #10 In= 1’b0;
      #10 In= 1’b1;
      #60

      $finish;

    end
endmodule
```

Simulación usando una descripción estructural instanciando flip-flops tipo D



Diseño usando una descripción comportamental.

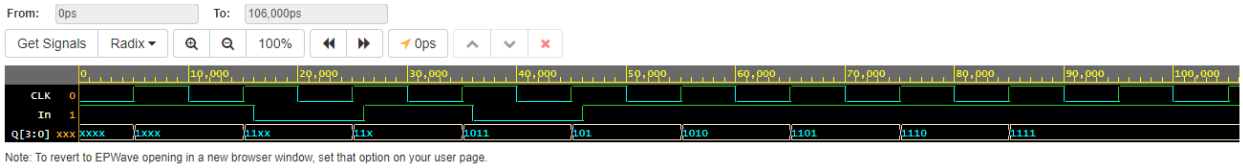
```
 ‘timescale  1ns / 10 ps
module registro_4bits(
  input CLK,
  output reg [3:0] Q,
  input In);

  always @(posedge CLK)
    Q = {In,Q[3:1]};
endmodule
```

Testbench usando una descripción comportamental.

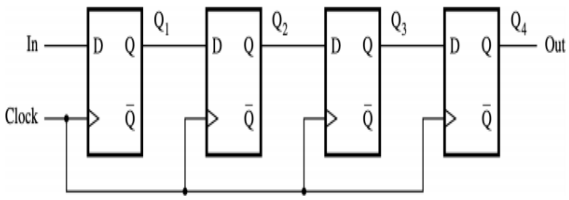
```
  ‘timescale  1ns / 10 ps

module test_registro_4bits ();
  reg CLK,In;
```

```
  wire [3:0] Q;

  registro_4bits circuito (CLK,Q,In);

  initial CLK = 1'b0;
  always #5 CLK = ~CLK;

  initial
    begin
      $dumpfile ("out.vcd");
      $dumpvars (1,test_registro_4bits);

      In =1'b1;
      #16 In= 1'b0;
      #10 In= 1'b1;
      #10 In= 1'b0;
      #10 In= 1'b1;
      #60
      $finish;
    end
endmodule
```

Simulación usando una descripción comportamental.



Registro de corrimiento simple de entrada serial y salida serial (SISO).
o Usando una descripción estructural instanciando flip-flops tipo D.
o Usando una descripción comportamental



Diseño Registro de corrimiento simple de entrada serial y salida serial (SISO).Usando una descripción estructural instanciando flip-flops tipo D

```
`timescale  1ns / 10 ps
module registro_siso(
  input CLK,
  output Out,
  input In);

  wire [3:0]Q;

```

```
  flipflop_D ff0 (In,Q[3],CLK);
  flipflop_D ff1 (Q[3],Q[2],CLK);
  flipflop_D ff2 (Q[2],Q[1],CLK);
  flipflop_D ff3 (Q[1],Q[0],CLK);

  assign Out= Q[0];
endmodule
```

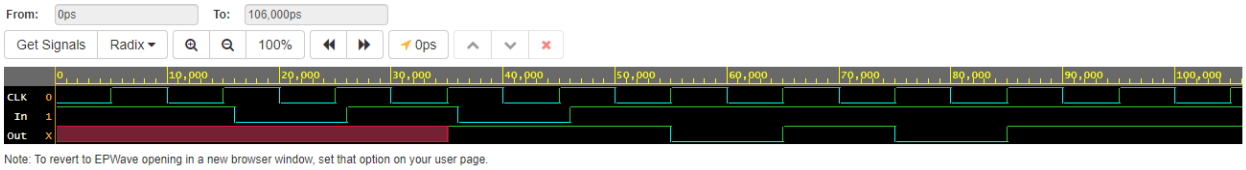
FlipFlop tipo D

```
`timescale  1ns / 10 ps
module flipflop_D(
  input D,
  output reg Q,
  input CLK);

  always @(posedge CLK)
    Q = D;
endmodule
```

Testbench Registro de corrimiento simple de entrada serial y salida serial (SISO).Usando una descripción estructural instanciando flip-flops tipo D

```

`timescale  1ns / 10 ps

module test_registro_siso ();
  reg CLK,In;
  wire Out;

  registro_siso circuito (CLK,Out,In);

  initial CLK = 1'b0;
  always #5 CLK = ~CLK;

  initial
    begin
      $dumpfile ("out.vcd");
      $dumpvars (1,test_registro_siso);

      In =1'b1;
      #16 In= 1'b0;
      #10 In= 1'b1;
      #10 In= 1'b0;
      #10 In= 1'b1;
      #60
      $finish;
    end
endmodule
```

Simulación Registro de corrimiento simple de entrada serial y salida serial (SISO).Usando una descripción estructural instanciando flip-flops tipo D



Note: To revert to EPWave opening in a new browser window, set that option on your user page.

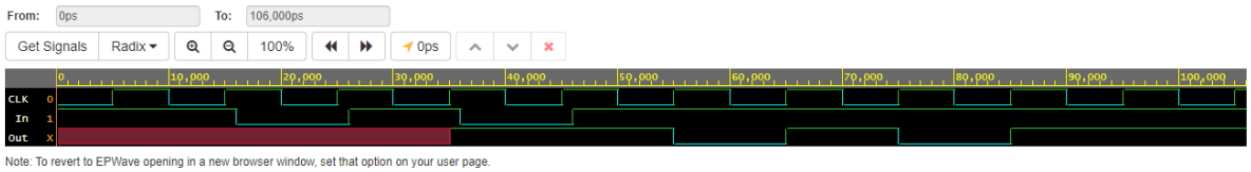
Diseño Registro de corrimiento simple de entrada serial y salida serial (SISO).Usando una descripción comportamental.

```
‘timescale  1ns / 10 ps

module registro_siso(
  input CLK,
  output Out,
  input In);

  reg [3:0]Q;

  always @(posedge CLK)
    Q = {In,Q[3:1]};

  assign Out = Q[0];
endmodule
```

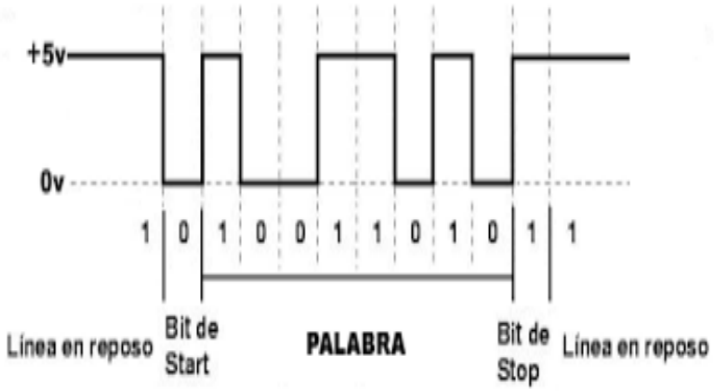
Testbench Registro de corrimiento simple de entrada serial y salida serial (SISO).Usando una descripción comportamental.

```
‘timescale  1ns / 10 ps

module test_registro_siso ();
  reg CLK,In;
  wire Out;
  registro_siso circuito (CLK,Out,In);

  initial CLK = 1’b0;
  always #5 CLK = ~CLK;

  initial
    begin
      $dumpfile ("out.vcd");
      $dumpvars (1,test_registro_siso);

      In =1’b1;
      #16 In= 1’b0;
      #10 In= 1’b1;
      #10 In= 1’b0;
      #10 In= 1’b1;
      #60
      $finish;
    end
endmodule
```

Simulación Registro de corrimiento simple de entrada serial y salida serial (SISO).Usando una descripción comportamental



Modulo RX protocolo RS232 (BAUD de 9600)
o Usando una descripción comportamental con las operaciones aritméticas y registros de SystemVerilog



Diseño Modulo RX protocolo RS232 (BAUD de 9600)

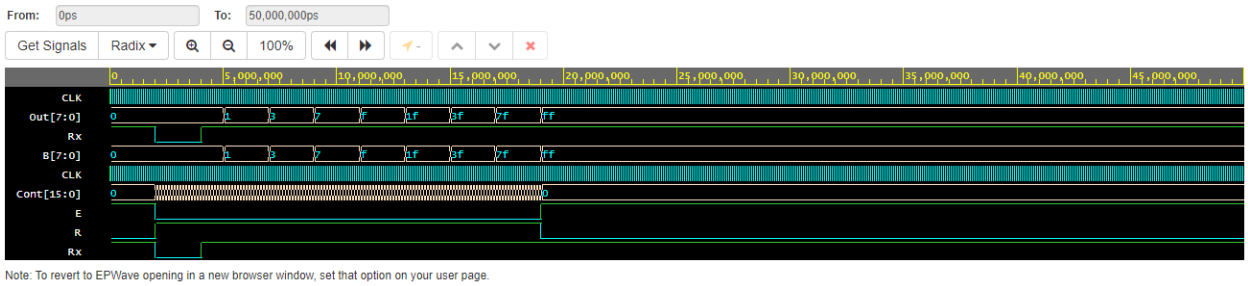
```
1  `timescale  1ns / 10 ps
2
3  module RS232(
4    input logic Rx,
5    input logic CLK,
6    output logic [7:0]B = 0);
7
8
9    logic [15:0]Cont = 0;
10   logic E = 1;
11   logic R = 0;
12
13   parameter ppb = 100;
14
15   always @(negedge CLK)begin
16     if(Rx == 0 && R == 0)begin
17       E = 0;
18       R = 1;
19     end
20
21     if (E == 0)begin
22       Cont = Cont + 1;
```

```
23
24      case (Cont)
25        (ppb*1 + (ppb/2)): B[0 ]= Rx;
26        (ppb*2 + (ppb/2)): B[1 ]= Rx;
27        (ppb*3 + (ppb/2)): B[2 ]= Rx;
28        (ppb*4 + (ppb/2)): B[3 ]= Rx;
29        (ppb*5 + (ppb/2)): B[4 ]= Rx;
30        (ppb*6 + (ppb/2)): B[5 ]= Rx;
31        (ppb*7 + (ppb/2)): B[6 ]= Rx;
32        (ppb*8 + (ppb/2)): begin
33          B[7] = Rx;
34
35          E = 1;
36          R = 0;
37          Cont = 0;
38        end
39      endcase
40    end
41  end
42 endmodule
```

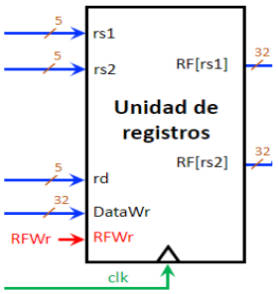
Testbench Modulo RX protocolo RS232 (BAUD de 9600)

```
1  module test;
2    logic CLK = 0;
3    logic Rx = 1;
4    logic [7:0]Out;
5
6    RS232 rs232 (Rx,CLK,Out);
7
8    parameter PERIODO = 20;
9    parameter ppb = 100;
10
11   always begin
12     #(PERIODO/2) CLK = ~CLK;
13   end
14
15   initial begin
16       $dumpfile ("out.vcd");
17       $dumpvars (2);
18     Rx= 1'b1;
19     #(PERIODO*ppb) Rx = 1'b0;
20     #(PERIODO*ppb) Rx = 1'b1;
21     #(PERIODO*ppb) Rx = 1'b1;
22     #(PERIODO*ppb) Rx = 1'b1;
23     #(PERIODO*ppb) Rx = 1'b1;
24     #(PERIODO*ppb) Rx = 1'b1;
25     #(PERIODO*ppb) Rx = 1'b1;
26     #(PERIODO*ppb) Rx = 1'b1;
27     #(PERIODO*ppb) Rx = 1'b1;
28     #(PERIODO*ppb) Rx = 1'b1;
29   end
30   initial
31     #(PERIODO *ppb*25) $finish;
32 endmodule
```

Simulación Modulo RX protocolo RS232 (BAUD de 9600)



Ejercicio Creación del módulo Unidad de Registros. El cual cuenta con los siguientes registros (cada uno con su respectivo número de bits): el ingreso de la dirección de dos registros (rs1, rs2), dos salidas con la información que fue solicitada (RF[rs1], RF[rs2]), activador de lectura (rd), activador de escritura(RFWr) e información de registro para ingresar(DataWrz).



Diseño Unidad de Registros

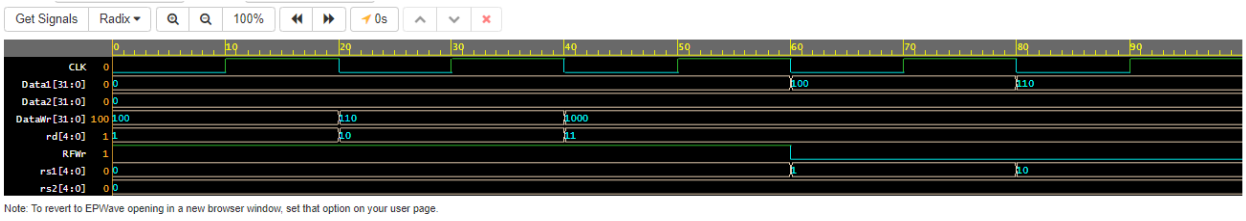
```
// Code your design here
module registerfile(
     input logic [4:0]rs1,
     input logic [4:0]rs2,
     input logic [4:0]rd,
     input logic [31:0]DataWr,
     input logic RFWr,
     input logic CLK,
     output logic [31:0] Data1,
     output logic [31:0]Data2 );

  logic [31:0] RF [31:0];

       initial
             begin
               $readmemh ("registros.txt",RF);
             end
  assign Data1 = RF [rs1];
  assign Data2 = RF [rs2];

  always @(posedge CLK)
         begin
           if(RFWr == 1)
```

```
            RF [rd] = DataWr;
        end
            endmodule
```

Testbench Unidad de Registros

```
    module Testebench;
logic [4:0] rs1 = 0;
logic [4:0] rs2 = 0;
logic [4:0]rd = 0;
logic [31:0]DataWr = 0;
logic RFWr= 0;
logic CLK;
logic [31:0]Data1;
logic [31:0]Data2;

     registerfile regfile (rs1,rs2,rd,DataWr,RFWr,CLK,Data1,Data2);

    parameter Period = 20;
          always
               begin
                      CLK = 1'b0;
                 #(Period/2);
                      CLK = 1'b1;
                 #(Period/2);
               end

     initial
            begin
              $dumpfile ("dump.vcd");
              $dumpvars (1);

                   RFWr = 1; rd = 1; DataWr = 4;
                   #20 RFWr = 1; rd = 2;DataWr = 6;
                   #20 RFWr = 1; rd = 3; DataWr = 8;
                   #20 RFWr = 0; rs1 = 1;
                   #20 RFWr = 0; rs1 = 2;
                   #20
              $finish();
            end
       endmodule
```

Simulación Unidad de Registros



Note: To revert to EPWave opening in a new browser window, set that option on your user page.

Procedimiento

Ejercicio 1: Flipflop Tipo D sensible al flanco de bajada de reloj usando una descripción comportamental

Se asignan tanto entradas como salidas(Incluyendo la entrada del reloj CLK).Se le asigna a la salida nQ la salida negada de Q y utlizando un proceso always, cuando el reloj se encuentra en un flanco de bajada (en negedge) se le asigna a la salida Q la entrada D.Se observa que por cada flanco de bajada, el valor correspondiente a la entrada D se pasa a la salida Q.

Ejercicio 2 :Flip-flop Tipo D sensible al flanco de subida de reloj con preset y clear activos en bajo usando una descripción comportamental.

A diferencia del anterior ejercicio, en este diseño se cuenta con un proceso always sensible a Preset y Clear, los cuales están activos en bajo cuando Preset se pone en O la salida Q toma el valor de 1, de lo contrario si Clear se pone en 0, la salida Q toma el valor de 0.

Ejercicio 3 :Registro de 4 bits Usando una descripción estructural instanciando flip-flops tipo D

Contamos con una entrada al reloj CLK, entrada de datos In y una slaida de 4 bits Q, se implementa un Flip Flop tipo de y se realiza la distanciación de 4 Flips Flops donde la salida de uno es la entrada de otro.Cada vez que hay un flanco de subida del reloj ingresa el valor de la entrada In a la salida de 4 bits

Ejercicio 4: Registro de 4 bits Usando una descripción comportamental.

En este caso se utiliza un proceso always a la salida Q se le concatena el bit que se encuentra en la entrada, con los tres bits más significativos en la entrada Q.

Ejercicio 5: Registro de corrimiento simple de entrada serial y salida serial (SISO). Usando una descripción estructural instanciando flip-flops tipo D

Se utiliza una entrada In y una salida serial Out, una entrada al reloj CLK, para esto utilizamos la implmentación de 4 Flip Flops realizados anteriormente, teniendo la salida Out el valor del último Flip Flop.

Ejercicio 6: Registro de corrimiento simple de entrada serial y salida serial (SISO) Usando una descripción comportamental.

Ejercicio 7: Módulo RX

Teniendo en cuenta que se necesitan 5208 ciclor del reloj por cada bit
Entrada al reloj CLK, entarda serial de datos y como salida un registro de 8 bist, un contador

de 4 bits para contar los bits de la palabra y una salida para el bit de parada y una salida para un codificador de contador. Instanciamos tres contadores y un registro sipo de 8 bits El módulo del contador tiene un registro muy parecido al de N bits.
Después de instanciar los contadores y asignar algunas pautas con el proceso always para que finalicen ciertos procesos, se inicia con el módulo de prueba .

Ejercicio 8: Unidad de registros
Se declara el módulo con los variables rs1 y rs2 de 5 bits, dos salidas Data1 y Data2 de 32 bits, un activador de lectura rd de 5 bits activador de escritura, una señal de reloj CLK y una variable para ingresar DAtaWrz de 32 bits.

De esta simulación se puede interpretar que es correcta ya que cuando hay un flanco de subida y el activador de escritura está en 1 se realiza el proceso de escritura. De forma asíncrona y el activador de lectura está en 0 se realiza el proceso de lectura

Problemas encontrados:

Realmente el tema tiene un nivel de complejidad mucho mayor en comparación con los anteiores laboratorios, para la relaización de los mismos fue indispensable la ayuda del monitor y la documentación de:

http://hyperphysics.phy-astr.gsu.edu/hbasees/Electronic/flipflop.html

Conclusiones:

Hasta ahora, los circuitos lógicos que se han considerado han sido combinatorios. En estos las salidas en cualquier punto del tiempo dependen completamente de las entradas que se presenten en ese momento. Aunque los circuitos combinatorios son la base para un gran número de aplicaciones, en la práctica la mayoría de los sistemas también incluyen elementos de almacenamiento, por lo que su análisis y diseño se debe realizar en términos de circuitos secuenciales.